Criteria for Polynomial Time (Conceptual) Clustering

Leonard Pitt
(pitt@a.cs.uiuc.edu)

Robert E. Reinke
(reinke@uicsl.csl.uiuc.edu)

Department of Computer Science
University of Illinois at Urbana-Champaign
1304 W. Springfield Ave.
Urbana, IL 61801 USA.

Running head: Polynomial Time Clustering Criteria

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
APR 2003		**00-00-2003 to 00-00-2003**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Criteria for Polynomial Time (Conceptual) Clustering	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Illinois at Urbana-Champaign,Department of Computer Science,1304 W. Springfield Avenue,Urbana,IL,61801**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**31**	

Abstract

Research in *cluster analysis* has resulted in a large number of algorithms and similarity measurements for clustering scientific data. Machine learning researchers have published a number of methods for *conceptual clustering*, in which observations are grouped into clusters which have "good" descriptions in some language. We investigate the general properties which similarity metrics, objective functions, and concept description languages must have to guarantee that a (conceptual) clustering problem is polynomial time solvable by a simple and widely-used clustering technique, the agglomerative-hierarchical algorithm. We show that under fairly general conditions, the agglomerative-hierarchical method may be used to find an optimal solution in polynomial time.

Keywords: Cluster Analysis, Conceptual Clustering, Analysis of Algorithms.

1 Introduction

There is a wide body of literature in several fields concerned with the *clustering problem*. Roughly, this is the problem of how to group observations into categories such that members of a category are alike in some interesting way and members of different categories are different. Within artificial intelligence and machine learning, the clustering problem has been classified as part of the general problem of learning from observation and discovery (Carbonell, Michalski, & Mitchell, 1983).

Much of the work on the clustering problem has involved numerical or statistical techniques for clustering scientific data. Researchers in *cluster analysis* and *numerical taxonomy* have focussed on developing appropriate metrics for measuring similarity between points and clusters (groups of points), and on developing algorithms to minimize inter-cluster similarity as measured by some objective function. The literature on these techniques is scattered through journals in statistics, pattern recognition, computer science, and various fields of application (biology, psychology, sociology, etc.). Summaries may be found in (Anderberg, 1973; Hartigan, 1975; Duda & Hart, 1973), and more recently (Romesburg, 1984).

In machine learning, work on the clustering problem has focussed on the notion of *conceptual clustering*, introduced by Michalski (1980). Conceptual clustering methods attempt not only to produce "good" classifications based on some metric, but also to find a meaningful description of the classification. In contrast, cluster analysis techniques leave it to the human analyst to determine the meaning of a clustering (here, and throughout the remainder of the paper, we use the term "cluster analysis" to refer to all clustering techniques in which the quality of cluster *descriptions* is not a factor in measuring the quality of the clustering). Researchers in conceptual clustering have not only produced a number of algorithms and metrics (e.g., Lebowitz, 1983; Michalski & Stepp, 1983; Fisher, 1985; Mogenson, 1987), but have also investigated employing clustering in problem solving (Rendell, 1983; Fisher, 1987), incorporating problem-specific knowledge into the clustering process (Mogenson, 1987; Stepp & Michalski, 1986), and providing appropriate cluster description languages (Stepp, 1984; Fisher, 1985). Fisher and Langley (1985) and Stepp (1987b) provide overviews of work on conceptual clustering, and develop characterizations of the problem.

Instead of adding to the already large body of clustering algorithms and metrics, we have explored the properties of a simple, general, and well-known clustering algorithm, the

agglomerative-hierarchical or amalgamative algorithm. In particular, we are interested in characterizing the kinds of problems, metrics, objective functions and concept description languages on which a given algorithm will succeed. We define success as the discovery of an optimal clustering in time polynomial in the number of data points to be clustered. Though introductory texts on cluster analysis sometimes attempt to explain how to choose an appropriate algorithm and metric, there has not, to our knowledge, been any formal exploration of the conditions under which particular algorithms are guaranteed to produce optimal solutions in polynomial time (there are, of course, algorithms designed to produce an optimal solution for particular metrics and objective functions). By specifying such conditions, we hope to simplify the problem of choosing an appropriate clustering technique.

In the next section, we introduce a simple conceptual clustering problem, and use it to motivate general definitions for clustering problems, distance metrics, and objective functions. In Section 3, the example is used to introduce the agglomerative algorithm, and to motivate some fairly general restrictions on conceptual clustering problems; in Section 4, these restrictions are proved sufficient to guarantee that the algorithm finds an optimal solution. We also examine further properties of the agglomerative algorithm, as well as variations of the restrictions of Section 3.

2 Definitions

We formally define a very general version of the clustering problem. The definitions are similar in spirit to the definition of the *Abstract Clustering Task* given by Fisher and Langley (1985). However, we want to more carefully specify what is meant by a "clustering quality function." We believe the definitions are general enough to subsume the objective functions normally used in cluster analysis and conceptual clustering.

We motivate the definitions, and the properties of the next section, with the following example of simplified *conjunctive conceptal clustering* (Michalski & Stepp, 1983), or *monomial clustering*. The formal definition of the monomial clustering problem will be given in Section 3.1.

For monomial clustering, the objects to be clustered are described with n boolean attributes $x_1, x_2, \ldots, x_n$. Let X_n be the set of boolean vectors over the attributes $x_1, x_2, \ldots, x_n$. Then the domain for the monomial clustering problem is $\mathcal{X} = \{X_n\}_{n \geq 1}$.

Similarly, many other domains for clustering problems are best described as a parameterized family $\mathcal{X} = \{X_n\}$ of sets, where the parameter n is some appropriate value, typically reflecting the "size" of an element. For example, if we are interested in clustering objects in Euclidean space, then the domain $\mathcal{X}$ might be the collection $\{E_n\}_{n\geq 1}$, where E_n is n-dimensional Euclidean space.

For some domains $\mathcal{X} = \{X_n\}$, we may allow X_n to be empty for some (or most, if no parameterization is desired) values of n. For example, if the only type of clustering problem that we wish to consider is clustering in 2-dimensional Euclidean space, then we would let $X_2 = E_2$, and $X_i = \emptyset$ for $i \neq 2$.

The above considerations motivate the following

Definition 2.1 *A* domain $\mathcal{X}$ *is a parameterized family of sets* $\{X_n\}_{n\geq 1}$.

Our definitions will require that a clustering algorithm work for all X_n that are nonempty, and that the algorithm have run-time polynomial in n.

In conceptual clustering, a cluster is described by a statement in some language, and not by the set of points in the cluster. For example, for monomial clustering, clusters are described by monomials over n boolean attributes. Let L_n be the set of all monomials (pure conjunctive concepts) over the boolean variables $x_1, x_2, \ldots, x_n$.

Definition 2.2 *A* clustering description language $\mathcal{L}$ *is a parameterized family of languages* $\{L_n\}_{n\geq 1}$.

The parameter n typically reflects the size or length of statements in the language L_n.

Each statement $c \in L_n$ is a *cluster*. The meaning of a cluster is given by an interpretation:

Definition 2.3 *An* interpretation $\mathcal{I} = \{I_n\}_{n\geq 1}$ *of a language* $\mathcal{L}$ *to a domain* $\mathcal{X}$ *is a parameterized family of functions* $I_n : L_n \rightarrow 2^{X_n}$. *($2^{X_n}$ is the power set of X_n.)*

For each n, every cluster $c \in L_n$ describes a set of points of X_n given by $I_n(c)$.

For monomial clustering, the interpretation I_n of a monomial over the variables x_1, x_2, $\ldots$, x_n is the standard logical one, i.e., the set of boolean vectors of length n (over the same variables) that satisfy the monomial. For some applications, it is desirable to parameterize the family of languages $\mathcal{L}$ differently than the domain $\mathcal{X}$. For example, if the goal is to cluster numerical data points based on their descriptions in a simple, limited language, then

for each n and m there would be an interpretation $I_{n,m} : L_n \rightarrow 2^{X_m}$. For the sake of clarity, we will assume that the language and its domain have the same parameter n; the extensions required for the two parameter case are straightforward.

A *clustering* C over L_n is a finite set of statements (clusters) of L_n. The *size* of a clustering C is the sum of the lengths of the statements in C. Let K_n denote the class of all clusterings over L_n. For monomials, a clustering is simply a finite collection of monomials.

A cluster c *covers* a set $S \subseteq X_n$ iff $S \subseteq I_n(c)$. A clustering C *covers* a set $S \subseteq X_n$ iff $S \subseteq \bigcup_{c \in C} I_n(c)$. The clustering C is a *prime clustering* of a set S iff C covers S and there is not a proper subset $C' \subset C$ such that C' covers S. A prime clustering is therefore one that contains no extraneous clusters.

The goal of clustering is to find, given a finite subset $S \subseteq X_n$ of elements, a (prime) clustering that covers S (and possibly other elements of X_n) such that each cluster of the clustering covers similar elements (i.e., is *tight*), and different clusters cover dissimilar elements (i.e., have large *distance*). The definitions of tightness and distance should depend only on clusters, i.e., on statements in the cluster description language, and not on the points covered by those clusters, thus fulfilling the primary condition of conceptual clustering (Michalski & Stepp, 1983; Fisher & Langley, 1985).

The tightness function will be parameterized by n in a manner similar to domains and clustering description languages. Since the clustering algorithm must employ this function, it is unreasonable to expect it to work for all n unless the tightness functions for each n are related to the extent that there is a single algorithm for evaluating them. For the same reason, the distance functions should be interrelated as well.

Definition 2.4 *A family $\mathcal{F} = \{F_n\}_{n \geq 1}$ of functions is* uniformly computable *iff there is an algorithm F such that for all n and x, $F(n, x) = F_n(x)$. A family is uniformly polynomial time computable iff F runs in time polynomial in the value of n and the size of x.*

The formal definition for tightness and distance functions are thus:

Definition 2.5 *$\mathcal{T} = \{T_n\}_{n \geq 1}$ is a uniformly computable family of* tightness *functions, with $T_n : K_n \rightarrow \Re^+$, where $\Re^+$ denotes the nonegative real numbers.*

Definition 2.6 *$\mathcal{D} = \{D_n\}_{n \geq 1}$ is a uniformly computable family of* distance *functions, with $D_n : K_n \rightarrow \Re^+$.*

T_n is a measure of tightness of clusterings over the language L_n. Note that T_n is a function of clusterings, and not of individual clusters. For monomial clustering, a natural measure of the tightness *of a monomial* is simply the number of attributes appearing in the monomial. A natural measure of overall tightness *of a clustering* (set of monomials) might be the minimum tightness of any monomial of the set. We define this tightness function $\mathcal{T}$ for monomials in the next section.

D_n is a measure of distance of clusterings over the language L_n. Note that D_n is a function of clusterings, not of pairs of clusters. In the next section, we define the distance D_n of a monomial clustering to be the minimum number of literals on which any pair of monomials of the clustering differ.

Numerical/statistical methods typically use a single metric that measures either similarity or dissimilarity between points. The objective function measures the overall quality of a clustering relative to that metric. For the sake of generality, we have assumed separate similarity (tightness) and dissimilarity (distance) measures, and allow the objective function (which we call "goodness") to be a function of these.

Definition 2.7 $\mathcal{G} = \{G_n\}_{n\geq 1}$ *is a uniformly computable family of* goodness *functions, with* $G_n : range(T_n) \times range(D_n) \to \Re^+$.

The goodness G_n of a clustering is a real number representing how well both tightness of clusters and distance between clusters has been achieved. We extend the domain of G_n to K_n with the natural interpretation that $(\forall C \in K_n)\ G_n(C) = G_n(T_n(C), D_n(C))$. We are now ready to define clustering problems.

Definition 2.8

- *A* (conceptual) clustering problem *is any six-tuple* $(\mathcal{X}, \mathcal{L}, \mathcal{I}, \mathcal{T}, \mathcal{D}, \mathcal{G})$ *with* $\mathcal{X}, \mathcal{L}, \mathcal{I}, \mathcal{T}, \mathcal{D}$, *and* $\mathcal{G}$ *defined as above.*

- *An* instance *of a conceptual clustering problem* $(\mathcal{X}, \mathcal{L}, \mathcal{I}, \mathcal{T}, \mathcal{D}, \mathcal{G})$ *is a seven-tuple* $(X_n, L_n, I_n, T_n, D_n, G_n,$ *where* $X_n \in \mathcal{X}$, $L_n \in \mathcal{L}$, $I_n \in \mathcal{I}$, $T_n \in \mathcal{T}$, $D_n \in \mathcal{D}$, $G_n \in \mathcal{G}$, *and* S *is any finite nonempty subset of* X_n.

- *The* solution *to an instance* $(X_n, L_n, I_n, T_n, D_n, G_n, S)$ *of a conceptual clustering problem is a clustering* $C \in K_n$ *(called a* best clustering*) such that*

1. *C is a prime clustering of S.*

2. *For all clusterings C' that satisfy 1. above, $G_n(C') \leq G_n(C)$.*

- *An algorithm A* solves *the conceptual clustering problem $(\mathcal{X}, \mathcal{L}, \mathcal{I}, \mathcal{T}, \mathcal{D}, \mathcal{G})$ iff for all n such that X_n is nonempty, and for any finite nonempty set $S \subseteq X_n$, the algorithm A, if given n and S as input, outputs a solution to the instance $(X_n, L_n, I_n, T_n, D_n, G_n, S)$. We also write that $(\mathcal{X}, \mathcal{L}, \mathcal{I}, \mathcal{T}, \mathcal{D}, \mathcal{G})$ is solvable.*

In what follows, the scope of the variable n will be all numbers such that X_n is nonempty. Thus the statement "for all n" is used to mean "for all n such that X_n is nonempty".

Note that if X_n is infinite, there may not exist a solution to $(X_n, L_n, I_n, T_n, D_n, G_n, S)$ because there may be an infinite sequence of clusterings for which G_n increases without bound. Also note that the solution (if it exists) of an instance of a clustering problem may not induce a partition of the points of S; there is no requirement that the clusters of the solution cover disjoint sets (of course, disjointness may be enforced by an appropriate choice of G_n). Further, the clusters may cover (possibly an infinite number of) points of $X_n - S$.

These definitions, and results in the following sections, are easily applied to the case of cluster analysis: The concept description language is simply finite subsets of X_n, and the interpretations $\mathcal{I}$ are identity functions I_n. Thus a cluster is simply a finite set of points of X_n.

Since we are interested in feasible computations, we define polynomial time solvability of clustering problems. Recall that the parameter n typically reflects a natural measure of size or length of encoding of objects $x \in X_n$.

Definition 2.9 *Let the families $\mathcal{T}, \mathcal{D}$, and $\mathcal{G}$ be uniformly polynomial time computable. Then $(\mathcal{X}, \mathcal{L}, \mathcal{I}, \mathcal{T}, \mathcal{D}, \mathcal{G})$ is* solvable in polynomial time *iff there is an algorithm A and polynomial p such that*

1. *A solves the clustering problem $(\mathcal{X}, \mathcal{L}, \mathcal{I}, \mathcal{T}, \mathcal{D}, \mathcal{G})$.*

2. *For any n (for which X_n is nonempty), and any finite $S \subseteq X_n$, the run-time of A on input n and S is at most $p(n, |S|)$.*

3 A Restricted Class of Clustering Problems

The definitions in Section 2 are so general that it would be ridiculous to expect that all clustering problems are polynomially solvable (or even solvable, for that matter). A main goal within our framework is to identify exactly those clustering problems that are (polynomially) solvable, and to give algorithms for solving them. Only by restricting the class of domains $\mathcal{X}$, the languages $\mathcal{L}$, and the objective functions $\mathcal{T}, \mathcal{D}$, and $\mathcal{G}$ under consideration, can we begin to make progress toward this goal. (As it turns out, we will not need to make any restrictions whatsoever on the domains $\mathcal{X}$.)

A natural algorithm for clustering is the following: Given a set of elements S to be clustered, begin by forming a cluster for each element of S. Then, iteratively, "merge" the two clusters which are "closest". Halt when there is only one cluster remaining (containing all of the points of S), and output the best clustering encountered during this process. This is essentially the agglomerative algorithm formally specified in Section 4.

In Section 3.1 we illustrate the agglomerative algorithm using an instance of the monomial clustering problem. In Section 3.2 the example is used to motivate properties for $\mathcal{L}$, $\mathcal{T}, \mathcal{D}$, and $\mathcal{G}$ that guarantee that the agglomerative algorithm finds an optimal solution.

3.1 An Example

The monomial clustering problem, discussed informally in the last section, is defined by:

- $\mathcal{X} = \{X_n\}_{n \geq 1}$, where X_n is the set of vectors over the variable set $x_1, x_2, \ldots, x_n$.
- $\mathcal{L} = \{L_n\}_{n \geq 1}$, where L_n is the set of monomials over the same variables (cf., the *single representation trick* (Cohen & Feigenbaum, 1983)).
- $\mathcal{I} = \{I_n\}_{n \geq 1}$, where I_n is the standard logical interpretation, i.e., the interpretation of a monomial is the set of boolean vectors that satisfy it.
- $\mathcal{T} = \{T_n\}_{n \geq 1}$, where, if $C = \{m_1, m_2, \ldots, m_k\}$ is a clustering of L_n,

$$T_n(C) \ = \ \min_{i=1,\ldots,k} \{t_n(m_i)\},$$

  and t_n, the tightness of a monomial, is the number of attributes (literals) in the monomial.

- $\mathcal{D} = \{D_n\}_{n \geq 1}$, where, if $C = \{m_1, m_2, \ldots, m_k\}$ is a clustering of L_n,

$$D_n(C) \;=\; \min_{1 \leq i \neq j \leq k} \{d_n(m_i, m_j)\},$$

  and d_n, the distance between two monomials, is the number of attributes which appear negated in one monomial and not negated in the other. If the clustering C has only one monomial, arbitrarily define $D_n(C) = 0$.

- $\mathcal{G} = \{G_n\}_{n \geq 1}$, where, if $C \in K_n$, then $G_n(C) = \min\{D_n(C), T_n(C)\}$.

The above objective functions $\mathcal{T}, \mathcal{D}$, and $\mathcal{G}$ capture the following three goals: (a) a tight clustering should contain only monomials that cover few points (a difference of 1 in the value of t_n corresponds to a factor of 2 in the number of points of X_n covered); (b) all monomials found should be disjoint (a clustering C containing non-disjoint monomials m_1 and m_2 will have $D_n(C) = 0$, since $d_n(m_1, m_2) = 0$), and should differ on as many attributes as possible; (c) a small value of T_n or D_n is equally undesirable, since the overall goal is to maximize the minimum of the two measures.

To see how the agglomerative algorithm works on the monomial clustering problem, we give a sample run using the instance (X_9, L_9, I_9, T_9, D_9, G_9, S), where the input set S consist of the points (events) $e_1, \ldots, e_5$ as follows:

$$e_1 = x_1\overline{x}_2x_3x_4x_5x_6x_7\overline{x}_8\overline{x}_9$$

$$e_2 = \overline{x}_1x_2\overline{x}_3\overline{x}_4\overline{x}_5x_6\overline{x}_7x_8x_9$$

$$e_3 = x_1\overline{x}_2x_3x_4\overline{x}_5\overline{x}_6x_7\overline{x}_8\overline{x}_9$$

$$e_4 = \overline{x}_1x_2\overline{x}_3\overline{x}_4\overline{x}_5\overline{x}_6x_7x_8x_9$$

$$e_5 = x_1\overline{x}_2x_3x_4x_5\overline{x}_6\overline{x}_7\overline{x}_8\overline{x}_9$$

For the rest of this section, we will refer to X_9, L_9, I_9, T_9, D_9, G_9, t_9, and d_9 as X, L, I, T, D, G, t, and d, respectively.

[Step 1]

The agglomerative algorithm begins with the clustering

$$C_1 = \{m_i = e_i : 1 \leq i \leq 5\}.$$

Note that this clustering is as "specific" as possible, in that each cluster is contained in some cluster of every monomial clustering that also covers S. The goodness of this clustering is 2, since all monomials have $t(m_i) = 9$, and the minimum distance (between the pairs (m_1, m_3), (m_1, m_5), (m_2, m_4), and (m_3, m_5)) is 2.

[Step 2]

The agglomerative algorithm chooses one of the minimally-distant pairs for merging. Assume it picks the pair (m_1, m_5). The obvious way to merge two monomials is to simply drop the attributes on which they differ. This results in a new clustering $C_2 = \{m_1, \ldots, m_4\}$ where:

$$m_1 = x_1\overline{x}_2x_3x_4x_5\overline{x}_8\overline{x}_9$$

$$m_2 = e_2 = \overline{x}_1x_2\overline{x}_3\overline{x}_4\overline{x}_5x_6\overline{x}_7x_8x_9$$

$$m_3 = e_3 = x_1\overline{x}_2x_3x_4\overline{x}_5\overline{x}_6x_7\overline{x}_8\overline{x}_9$$

$$m_4 = e_4 = \overline{x}_1x_2\overline{x}_3\overline{x}_4\overline{x}_5\overline{x}_6x_7x_8x_9$$

The new cluster m_1 covers the events e_1 and e_5, as well as other points of X_n. Its tightness t is 7, and the new minimum distance (between m_1 and m_3) is 1. Therefore, the goodness of the new clustering is also 1. Note that this is less than the goodness of the initial clustering; the algorithm does not hill-climb on this objective function.

[Step 3]

The algorithm merges the monomials m_1 and m_3. This results in a new clustering $C_3 = \{m_1, m_2, m_3\}$ where:

$$m_1 = x_1\overline{x}_2x_3x_4\overline{x}_8\overline{x}_9$$

$$m_2 = e_2 = \overline{x}_1x_2\overline{x}_3\overline{x}_4\overline{x}_5x_6\overline{x}_7x_8x_9$$

$$m_3 = e_4 = \overline{x}_1x_2\overline{x}_3\overline{x}_4\overline{x}_5\overline{x}_6x_7x_8x_9$$

[Step 4]

The minimum tightness is 6 (m_1) and the minimum distance is 2 (between m_2 and m_3). The algorithm therefore merges m_2 and m_3, resulting in $C_4 = \{m_1, m_2\}$ where:

$$m_1 = x_1\overline{x}_2x_3x_4\overline{x}_8\overline{x}_9$$

$$m_2 = \overline{x}_1 x_2 \overline{x}_3 \overline{x}_4 \overline{x}_5 x_8 x_9$$

This clustering has $T = 6$ and $D = 6$, so $G = 6$. This is, in fact, a best clustering of the events under the given objective function.

[**Step 5**]

The last step in the example merges the two remaining clusters into a single monomial to obtain the clustering $C_5 = \{\emptyset\}$ with with $G(C_5) = T(C_5) = D(C_5) = 0$. The algorithm therefore keeps C_4 as the best clustering.

3.2 Properties

In this section, we present some properties for clustering problems, using the example from the previous section to provide intuitive motivations. For a clustering problem $(\mathcal{X}, \mathcal{L}, \mathcal{I}, \mathcal{T}, \mathcal{D}, \mathcal{G})$, we state only the (more restrictive) properties sufficient for polynomial time solvability; corresponding properties for general solvability are trivially obtained by dropping the polynomial time requirements. In Section 4, we will prove that a clustering problem possessing the properties is solvable in polynomial time because the agglomerative algorithm meets the requirements of Definition 2.9.

In our example, the cluster description language and interpretation make it is easy to determine whether a point $x \in X_n$ is covered by a statement $c \in L_n$. The ability to determine cluster membership is necessary if the agglomerative algorithm is to create prime clusterings – the merging operation used in the agglomerative algorithm may produce a new cluster whose interpretation is a superset of (the interpretation of) some cluster not involved in the merge. To guarantee prime clusterings, we must be able to detect such extraneous clusters. The first property therefore requires that we be able to determine cluster membership in polynomial time.

Property P_1: There exists a polynomial time algorithm such that when given as input any number n, any point $x \in X_n$, and any cluster $c \in L_n$, outputs "true" if $x \in I_n(c)$, and "false" otherwise.

The membership algorithm may be used to obtain a prime clustering from a given clustering of a set S. In particular, let the polynomial time subroutine PRIME(n, C, S) return a

prime clustering C' of S, where C' contains a subset of the clusters of C. PRIME iteratively examines each cluster $c \in C$ and adds it to C' iff there is some point $x \in S$ such that $x \in I_n(c)$ and C' does not cover $\{x\}$.

The next property is based on step 1 of the example. In this step, the algorithm created a single cluster for each point $x_i \in S$. Each cluster c_i covered x_i and as few additional points of X as possible. For some languages L_n and some sets S, there may not exist a clustering of individual points which is "most specific" in this sense. Property P_2 asserts that the cluster description languages must be such that for any set of points in X_n, there must exist, and there must exist feasible (polynomial) means for finding, a cluster (statement in L_n) that is the most specific of any statement in L_n covering those points.

Definition 3.1 *For a given clustering problem* $(\mathcal{X}, \mathcal{L}, \mathcal{I}, \mathcal{T}, \mathcal{D}, \mathcal{G})$*, and any* n*, the* maximally specific cover (MSC) *for a set of points* $P \subseteq X_n$ *is a cluster* $c \in L_n$ *such that* c *covers* P*, and for any* $c' \in L_n$*, if* c' *covers* P*, then* $I_n(c) \subseteq I_n(c')$.

Property P_2: There exists an algorithm such that when given as input any number n and finite $S = \{x_1, x_2, \ldots x_s\} \subseteq X_n$, outputs a clustering $C = \{c_1, c_2, \ldots, c_s\} \in K_n$ such that for $1 \leq i \leq s$, c_i is an MSC for $\{x_i\}$. The run-time of the algorithm must be polynomial in n and $|S|$.

Property P_2 implies that the descriptions $\{c_i\}$ of the maximally specific covers of the singleton point sets be at most of size polynomial in the total size of S (otherwise, the algorithm may spend exponential time just creating them).

The next two properties are based on the merging operation in the example. The agglomerative algorithm assumes that $D(C)$ for a clustering is based on an inter-*cluster* distance measure d:

Property P_3: There is a family of uniformly polynomial time computable functions $d = \{d_n\}$, where $d_n : L_n \times L_n \rightarrow \Re^+$, such that for all n, and $C \in K_n$,

(a) $D_n(C) = min\{d_n(c_i, c_j) : c_i, c_j \in C, c_i \neq c_j\}$.
When restricted to single-cluster clusterings, $\{D_n\}$ may be any family of uniformly polynomial time computable functions satisfying (b) below.

(b) $(\forall c_1, c_2, c_3, c_4 \in L_n)$ $(I_n(c_1) \subseteq I_n(c_2))$ and $(I_n(c_3) \subseteq I_n(c_4)) \Rightarrow d_n(c_1, c_3) \geq d_n(c_2, c_4)$.

If $C_1 = \{c_1\}$ and $C_2 = \{c_2\}$ are clusterings containing only single clusters, then $I_n(c_1) \subseteq I_n(c_2) \Rightarrow D_n(C_1) \geq D_n(C_2)$.

Property P_3 requires that the distance D of a clustering really is the minimum inter-cluster "distance" d between any pair of clusters, where d has a monotone property: If two clusters have distance d, and points are then added to each, the distance d cannot increase. In other words, as clusters "grow", the distance between them shrinks, and D is the minimum of all these inter-cluster distances.

It is not clear what is meant by the "distance" of a clustering C when C contains only a single cluster. Generally, one is only interested in clusterings that contain more than one cluster. A possible way to deal with this is to simply let the value of G be zero for any such clustering, or to let the value of D be zero. For the sake of generality, we have chosen the weakest requirement, which is to allow D to be defined arbitrarily for one-cluster clusterings, but to require that D be monotone under generalization. Perhaps more natural, but also more restrictive, would be to require that the value of D be the same for all one-cluster clusterings, or to let G be defined as a function of $T(C)$ alone when C has only one cluster.

In steps 2 – 4 of the example, the algorithm merged monomials in an obvious way to produce new clusters. Also, the clusters produced by merging monomials were always maximally specific covers (for the points covered by the merged clusters). Property P_4 is related to P_2. It requires that we be able to generate, in polynomial time, an MSC for the union of any two sets of points in the problem space described by clusters:

Property P_4: There is an effective procedure M such that for any n, M "merges" any two clusters $c, c' \in L_n$. For all n, M and L_n must have the following properties:

(a) For all $c, c' \in L_n$, there is an MSC c'' for $I_n(c) \cup I_n(c')$, and $M(n, c, c') = c''$.

(b) M runs in polynomial time, i.e., in time polynomial in n, and in the lengths of the statements c and c'.

(c) There is a polynomial q such that for any finite subset S of X_n, if c is obtained by any (finite) number of merges of MSCs of subsets of S, then c has size at most $q(n, |S|)$.

Part (c) assures that the following event does not occur: During a run of the agglomerative algorithm, clusters are "merged" successively. It is possible that at some point, the description of a cluster is larger than some polynomial in the size of S. Any reasonable restriction on the size of the statement $M(n, c, c')$ in part (b) will not prevent this event from occurring. For example, since there are at most $|S|$ iterations, even if we require that the length of the statement $M(n, c, c')$ is at most the sum of the lengths of the statements c and c', it is possible that the final clustering obtained by the algorithm will have size exponential in $|S|$. By requiring part (c) in addition to part (b), we guarantee that any statement produced by the algorithm has size at most polynomial in the size of S.

In many cases (e.g., when clusters are conjunctive descriptions over any collection of attributes), the size of descriptions will *decrease* as clusters become more general. In other cases (e.g., axis-aligned rectangles in Euclidean spaces), description size will remain constant. In cluster analysis, property P_4 is trivially satisfied, since merging is done by union (of sets of points), and the largest statement is exactly S.

Property P_5 is based on the observation that, in the monomial example, clusters became less cohesive (more general or less *tight*) as merges occurred. We need the following definition:

Definition 3.2 *Given X_n, L_n, and I_n, the relation $\trianglelefteq_n$ on K_n is defined by: For all $C, C' \in K_n$, $C \trianglelefteq_n C'$ iff $(\forall c \in C)(\exists c' \in C')\ I_n(c) \subseteq I_n(c')$. If $C \trianglelefteq_n C'$ we say C is less general than, more specific than, and is a specialization of, C', and equivalently, that C' is more general than, less specific than, and is a generalization of, C. When restricted to prime clusterings with respect to a given set S, $\trianglelefteq_n$ is a partial order on K_n.*

Property P_5: The tightness functions $\mathcal{T} = \{T_n\}$ are a uniformly polynomial time computable family of functions, and for all n, the function T_n is *monotone nonincreasing under generalization*, i.e., for all $C, C' \in K_n$, if $C \trianglelefteq_n C'$ then $T_n(C) \geq T_n(C')$.

Property P_5 asserts that if one clustering is a generalization of another, then the more general clustering is at most as tight as the less general. Because we have defined tightness as a function of clusterings, and not of individual clusters, it is not immediately clear that this is a natural property. Observe, however, that the definition of the relation $\trianglelefteq_n$ states that each cluster of the less general clustering is contained in some cluster of the more general clustering. Thus if T_n somehow depends on the "tightness" of particular clusters (e.g., if tightness of individual clusters is inversely related to the quantity or variety of elements

covered), then the more general clustering contains clusters at most as tight as the clusters of the less general clustering. We would then expect that the overall value of T_n would be more for the less general clustering.

Finally, property P_6 simply says that goodness has a very natural property: If you increase either distance or tightness, while holding the other constant, then goodness should not decrease. In other words, tight, distant clusterings are best.

Property P_6: The goodness functions $\mathcal{G} = \{G_n\}$ are a uniformly polynomial time computable family of functions, and for all n, the function G_n is *monotone nondecreasing in T_n and D_n*. That is, if $x_1 \geq x_2 \in range(T_n)$ and $y_1 \geq y_2 \in range(D_n)$, then $G_n(x_1, y_1) \geq G_n(x_2, y_1)$ and $G_n(x_1, y_1) \geq G_n(x_1, y_2)$.

3.3 Example Clustering Problems

The properties P_1 through P_6 hold for several interesting conceptual clustering and cluster analysis problems that fall within our framework. In this section, we present some of these, without proof that they do indeed satisfy the properties.

Monomials The properties P_1 – P_6 were motivated by, and are natural generalizations of, properties held by the monomial clustering problem. It is easily verified that these properties are satisfied by the monomial clustering problem as defined in Section 3.1. The properties also hold for conjunctive conceptual clustering using multiple-valued attributes and internal disjunction (Michalski & Stepp, 1983).[1] In this case, the initial clustering is as for monomials and the "refunion" operator (Michalski, 1983) can be used for merging. Natural extensions of the distance and tightness measures in the example satisfy P_3 and P_5, and these may be used with any objective function satisfying P_6.

Geometric Another interesting group of languages which have these properties are some geometric languages over Euclidean spaces. For example, the agglomerative algorithm can solve the *axis-aligned rectangle clustering problem*, defined by:

- $\mathcal{X} = \{X_n\}$, where X_n is n-dimensional Euclidean space.

[1] Some of the metrics used by Michalski & Stepp, e.g., "simplicity" and "sparseness", clearly do not satisfy properties P_3 and P_5.

- $\mathcal{L} = \{L_n\}$, where L_n is the set of n-dimensional rectangles with sides parallel to the n axes.
- $\mathcal{I} = \{I_n\}$ is the standard interpretation: $I_n(r)$, where r is a rectangle of L_n, is the set of points of X_n that are contained in r.
- $\mathcal{T} = \{T_n\}$, where $T_n(\mathrm{C})$ for a collection of rectangles C could be any of
  1. The inverse of the area of the union of the rectangles of C.
  2. The inverse of the area of the largest rectangle of C.
  3. The inverse of the maximum distance between any two points *within* any cluster, i.e., the inverse of the length of the longest diagonal.
- $\mathcal{D} = \{D_n\}$, where $D_n(\mathrm{C})$ is the minimum pairwise "distance" d_n between any pair of rectangles of C, and d_n is any metric that gets smaller as clusters grow. (Some metrics d_n that *do not* have this property include $d_n(r_1, r_2) =$ maximum distance between any pair of points of r_1 and r_2, or distance between the centers of r_1 and r_2.) Let $D_n(C) = 0$ if C has only one cluster.
- $\mathcal{G} = \{G_n\}$ is any objective function satisfying P_6.

The critical condition for geometric languages is that there exist, and we can find, a description of the smallest set representable in the language that covers a given set of points. For example, if the language consists of descriptions of all convex polygons in 2-dimensional Euclidean space, then it is easy to see that the agglomerative algorithm may be successfully applied, since the convex hull of a set of points (which may be found in polynomial time) is contained in every convex polygon containing the points. However, if $\mathcal{L} = \{L_n\}$, where L_n consists of descriptions of convex polytopes in n dimensions (i.e., a list of $(n-1)$-dimensional hyperplanes), then property P_2 (and P_4) do not hold, because the length of a description of the convex hull of a set of s points in n dimensions (i.e., the length of the description of the MSC of a set of points) can be as large as $s^{\lfloor \frac{n}{2} \rfloor}$ (Edelsbrunner, 1987). If we are willing to relax the requirement that the clustering found have size polynomial in the dimension, then the agglomerative algorithm can be used to find an optimal clustering. The MSCs are obtained by applying any algorithm for finding the convex hull of a set of points in n dimensions. Alternatively, by using a different representation of the convex hull

(e.g., the extremal points), then properties P_1 - P_6 will hold, and the agglomerative algorithm may be used to find optimal clusters in time polynomial in the dimension and the initial number of points.

Cluster Analysis Within our framework, any cluster analysis problem trivially satisfies properties P_1, P_2, and P_4. Whether properties P_3, P_5, and P_6 are satisfied will depend on the particular choice of the objective functions. Single linkage clustering (Anderberg, 1973), in which the distance between clusters (d_n in our framework) is the minimum distance between points in the clusters, clearly satisfies the monotone requirement for distance metrics in property P_3. Complete linkage clustering (d_n is the *maximum* inter-point distance) clearly does not. Most reasonable choices of $\mathcal{G}$ will satisfy property P_6.

4 The Agglomerative Algorithm

4.1 Algorithm A

The algorithm we consider is the *Central Agglomerative Procedure* as described by Anderberg (1973). Variants of this method constitute the majority of the work on *hierarchical* clustering (Romesburg, 1984). Hierarchical clustering techniques are normally used to produce a classification tree over the object set, where leaves are individual objects, and internal nodes represent clusters. We will instead be concerned with whether the technique finds a single clustering which is best under the objective function. In this sense, we are using the agglomerative/hierarchical procedure as an optimization technique (Everitt, 1980), but without fixing the number of clusters beforehand. We are also allowing the algorithm to produce non-disjoint clusters (cf., *clumping* techniques (Everitt, 1980)).

Given n and a finite nonempty set $S \subseteq X_n$, the algorithm produces $t \leq |S|$ different clusterings C_1, $C_2, \ldots, C_t$, by starting with the maximally specific cluster for each point in S and successively merging clusters with minimum distance until a single cluster covering all of S is obtained. After each merge, extraneous clusters are eliminated. The output of the algorithm is the clustering among $C_1, C_2, \ldots C_t$ with the best value. We will prove that the algorithm solves any clustering problem $(\mathcal{X}, \mathcal{L}, \mathcal{I}, \mathcal{T}, \mathcal{D}, \mathcal{G})$ that satisifes properties P_1 through P_6. (The algorithm itself implicitly assumes that properties P_1, P_2, P_3 part (a), and

P_4 hold.)

Agglomerative Algorithm A

INPUT(n, S)
FOR each $x_i \in S$, compute c_i, an MSC for $\{x_i\}$
$C_1^{temp} \leftarrow \{c_i : x_i \in S\}$
$C_1 \leftarrow$ PRIME(n, C_1^{temp}, S)
IF $|C_1| = 1$ THEN done $\leftarrow$ TRUE, ELSE done $\leftarrow$ FALSE
i $\leftarrow$ 1
WHILE done $\neq$ TRUE DO BEGIN
    i $\leftarrow$ i+1
    compute $d_n(c_j, c_k)$ for each $c_j, c_k \in C_{i-1}$
    let $c, c' \in C_{i-1}$ be such that $D_n(C_{i-1}) = d_n(c, c')$
        (c and c' are the two closest clusters of C_{i-1}.)
    $C_i^{temp} \leftarrow C_{i-1} - \{c, c'\} \cup M(n, c, c')$
        (C_i^{temp} is C_{i-1} with clusters c and c' merged.)
    $C_i \leftarrow$ PRIME(n, C_i^{temp}, S)
        (eliminate any extraneous clusters.)
    IF $|C_i| = 1$ THEN done $\leftarrow$ TRUE
END
$t \leftarrow i$ (index of final clustering formed)
OUTPUT any $C \in \{C_1, \ldots C_t\}$ such that $G_n(C)$ is maximum.

Theorem 4.1 *If $(\mathcal{X}, \mathcal{L}, \mathcal{I}, \mathcal{T}, \mathcal{D}, \mathcal{G})$ is any clustering problem such that properties P_1 through P_6 are satisfied, then A solves $(\mathcal{X}, \mathcal{L}, \mathcal{I}, \mathcal{T}, \mathcal{D}, \mathcal{G})$ in polynomial time.*

Proof: By property P_2, the clustering C_1^{temp} is found in polynomial time. Note that $d = \{d_n\}$, $\mathcal{D} = \{D_n\}$, $\mathcal{T} = \{T_n\}$, and $\mathcal{G} = \{G_n\}$ are uniformly polynomial time computable families (properties P_3, P_5, and P_6), and that the merge operation M never produces a statement of length greater than $q(n, |S|)$ (property P_4 parts (b) and (c)). Subroutine PRIME runs in polynomial time by the comments following the introduction of property P_1. Since there are $t \leq |S|$ iterations (because $|C_{i+1}| < |C_i|$), the algorithm runs in time polynomial in $|S|$ and n. We need only show that the algorithm is correct.

Lemma 4.2 *Let $(X_n, L_n, I_n, T_n, D_n, G_n, S)$ be an instance of a clustering problem that satisfies properties P_1 through P_6. Let $C \in K_n$ be a prime clustering of S, a specialization of some best clustering, and suppose that C itself is* not *a best clustering. Let $c, c' \in C$ be such that $D_n(C) = d_n(c, c')$, and let $C' = C - \{c, c'\} \cup M(n, c, c')$. (In other words, C' is obtained from C by merging two clusters with minimum distance d_n.) Then C' is a specialization of a best clustering, as is $PRIME(n, C', S)$.*

We first show that Theorem 4.1 follows from Lemma 4.2, and then prove Lemma 4.2. To prove the theorem, we need only show that at least one of the clusterings $\{C_1, C_2, \ldots, C_t\}$ is a best clustering.

Suppose by way of contradiction that none of the C_i's is a best clustering. By the definition of the maximally specific cover (MSC) of a set of points P, any cluster which covers P must cover a superset of the points covered by the MSC. C_1^{temp} consists of the MSCs for each point in S. Thus, C_1^{temp} is a specialization of every clustering of S and therefore of a best clustering. Trivially, C_1 is a specialization of a best clustering, and is a prime clustering of S. By Lemma 4.2 (and the definition of C_2^{temp}), C_2^{temp} is a specialization of a best clustering, as is C_2. Iteratively applying Lemma 4.2 and our supposition that none of the C_i's are best, we have that each of $C_1, C_2, \ldots, C_t$ is a specialization of a best clustering. (Each is also a prime clustering of S.) But this is a contradiction, for C_t cannot be a specialization of a best clustering without being a best clustering: Since C_t has only one cluster, and only prime clusterings are candidate solutions, any generalization C_{best} of C_t must have exactly one cluster that contains the single cluster of C_t. Further, $T_n(C_{best}) \leq T_n(C_t)$ and $D_n(C_{best}) \leq D_n(C_t)$ by properties P_5 and P_3 part (b). By P_6, $G_n(C_{best}) \leq G_n(C_t)$, and thus C_t is in fact a best clustering. It follows that our supposition was wrong, and at least one of $\{C_1, C_2, \ldots, C_t\}$ must be a best clustering. □

We now prove Lemma 4.2. Let C_{best} be a best clustering, with $C \trianglelefteq_n C_{best}$, and C a prime clustering of S, but not a best clustering. Then $G_n(C) < G_n(C_{best})$.

Let c, c', and C' be as defined in the lemma. (Thus $d_n(c, c') = D_n(C)$.) Since $C \trianglelefteq_n C_{best}$, there are clusters $b, b' \in C_{best}$ such that $I_n(c) \subseteq I_n(b)$ and $I_n(c') \subseteq I_n(b')$. There are now two cases:

Case 1: $b = b'$

Then the only cluster of C' that is not also a cluster of C is $M(n, c, c')$. Observe

that $I_n(c) \cup I_n(c') \subseteq I_n(b)$, and by the definition of $M(n, c, c')$ as maximally specific, $I_n(M(n, c, c')) \subseteq I_n(b)$. Thus C' is a specialization of C_{best}. Trivially, PRIME(n, C', S) is also a specialization of C_{best}, and the lemma is proved.

Case 2: $b \neq b'$

In this case, $T_n(C) \geq T_n(C_{best})$ by property P_5, and now note that:

$$\begin{aligned} D_n(C) &= d_n(c, c') && \text{(by choice of } c, c') \\ &\geq d_n(b, b') && \text{(by property } P_3 \text{ part (b))} \\ &\geq D_n(C_{best}) && \text{(by property } P_3 \text{ part (a)).} \end{aligned}$$

Since $T_n(C) \geq T_n(C_{best})$ and $D_n(C) \geq D_n(C_{best})$, by property P_6, $G_n(C) \geq G_n(C_{best})$, contradicting the hypothesis of the lemma that C is not a best clustering. Thus case 1 *must* hold, completing the proof of Lemma 4.2 and Theorem 4.1. □

4.2 Properties of Algorithm A

It is interesting to note that algorithm A is *not* a hill-climbing algorithm, in that the value of the objective function may increase and decrease as the sequence of clusters $C_1, C_2, \ldots, C_t$ is formed. (Recall the monomial example in Section 3.1.) It is true, however, that the measure of tightness T_n is monotone nonincreasing as each new clustering is examined. The function D_n is not necessarily monotone, because it is possible for D_n to increase when two clusters are merged (since the minimum distance is eliminated), and also to decrease (since the new larger cluster may be very close to some other cluster). It is for this reason that the algorithm must continue generating clusterings rather than stopping once the value of G_n decreases. Under some objective functions, the algorithm may hill-climb. Single-linkage cluster analysis problems (Anderberg, 1973), for example, define d_n as the minimum "distance" between points of two clusters (where "distance" is any metric). If such a problem satisfies properties P_1 - P_6, then the agglomerative algorithm will hill-climb on the objective function.

It is also worth noting that algorithm A finds, for each $k \leq s$, the best clustering with *at least* k clusters. We will show that for $1 \leq k \leq s$, the best clustering with at least k clusters is in the set $\{C_1, C_2, \ldots, C_t\}$. This is achieved by proving, for each fixed $k \leq s$, the following variant of Lemma 4.2.

Let $best_k$ mean "best among all prime clusterings of S with at least k clusters".

Lemma 4.3 *Let $(X_n, L_n, I_n, T_n, D_n, G_n, S)$ be an instance of a clustering problem that satisfies properties P_1 through P_6. Let $C \in K_n$ be a prime clustering of S containing at least k clusters. Further, let C be a specialization of some best$_k$ clustering, and suppose that C itself is* not *a best$_k$ clustering. Let $c, c' \in C$ be such that $D_n(C) = d_n(c, c')$, and let $C' = C - \{c, c'\} \cup M(n, c, c')$. (In other words, C' is obtained from C by merging two clusters with minimum distance d_n.) Then C' is a specialization of a best$_k$ clustering, as is $PRIME(n, C', S)$.*

Lemma 4.3 differs from Lemma 4.2 only in that "best clustering" has been replaced with "*best$_k$* clustering" and the additional hypothesis that C has at least k clusters has been added. The proof of Lemma 4.3 is nearly identical to the proof of Lemma 4.2. (One needs the fact that C has at least k clusters to arrive at the contradiction in Case 2.)

We can now prove

Theorem 4.4 *If $(\mathcal{X}, \mathcal{L}, \mathcal{I}, \mathcal{T}, \mathcal{D}, \mathcal{G})$ is any clustering problem such that properties P_1 through P_6 are satisfied, then for each $k \leq |S|$, the set of clusterings $\{C_1, C_2, \ldots, C_t\}$ produced by algorithm A contains a best$_k$ clustering (if one exists).*

To prove Theorem 4.4, we assume that for some $k \leq |S|$, a *best$_k$* clustering exists (one could fail to exist because every prime clustering could have fewer than k clusters), and that none of the (prime) clusterings $\{C_1, C_2, \ldots, C_t\}$ is a *best$_k$* clustering. We then obtain a contradiction.

Consider the sequence of clusterings $C_1^{temp}, C_1, C_2^{temp}, C_2, \ldots, C_t^{temp}, C_t$, produced during the run of algorithm A. Since $C = C_1^{temp}$ satisfies

(a) C has at least k clusters

(b) C is a specialization of a *best$_k$* clustering,

there is a rightmost element R of this sequence of clusterings that satisfies (a) and (b). There are 3 cases, each resulting in a contradiction:

Case 1: For some i, $1 \leq i < t$, $R = C_i$. Then, since R is a prime clustering of S, by (a), (b), our assumption that none of the C_i's is a *best$_k$* clustering, and Lemma 4.3, we conclude that C_{i+1}^{temp} is a specialization of a *best$_k$* clustering. Then C_{i+1}^{temp} must have less than k clusters, otherwise $R = C_{i+1}^{temp}$ instead of C_i. Since C_i and C_{i+1}^{temp} differ

in number of clusters by exactly one, it follows that C_i has exactly k clusters. Let C_{best} be a (prime) generalization of C_i that is a $best_k$ clustering. Then C_{best} must have *exactly* k clusters, each a superset of a different cluster of C_i. Thus $T_n(C_i) \geq T_n(C_{best})$, $D_n(C_i) \geq D_n(C_{best})$, and $G_n(C_i) \geq G_n(C_{best})$, contradicting the assumption that C_i is not a $best_k$ clustering.

Case 2: $R = C_t$. Then since C_t has exactly one cluster, $k = 1$. In other words, C_t has exactly k clusters, and the reasoning concluding case 1 above may be employed.

Case 3: For some i, $1 \leq i \leq t$, $R = C_i^{temp}$. Since C_i^{temp} is a specialization of some $best_k$ clustering C_{best}, so is C_i. Then C_i must have less than k clusters, otherwise $R = C_i$. Both C_i and C_{best} are prime clusterings of S, so C_{best} can have at *most* the same number of clusters as C_i (one superset of each $c \in C_i$). Therefore, C_{best} has less than k clusters, and is not a $best_k$ clustering, a contradiction.

Since in each case we have arrived at a contradiction, our assumption that none of the clusterings $\{C_1, C_2, \ldots, C_t\}$ is a $best_k$ clustering must be false, completing the proof of Theorem 4.4. □

A natural question is whether it is possible to find a best clustering with *exactly* k clusters. Certainly this is at least as difficult as finding a best clustering with *at most* k clusters, since an algorithm for the former problem could be run k times to find best clusterings with exactly $1, 2, 3, \ldots, k$ clusters, and the best could be chosen as an answer to the latter problem. We show that there exists a clustering problem satisfying P_1 - P_6 such that unless $P = NP$, no polynomial time algorithm is guaranteed to find, for all instances of the problem, a best clustering with at most k clusters.

It would appear that a simple reduction from the NP-hard CLUSTERING problem (Garey & Johnson, 1979) would be sufficient to show this. However, due to the definition of what an "instance" of each problem is, a straightforward approach relating the "distance" function of CLUSTERING to any of our measures $\mathcal{T}, \mathcal{D}$, or $\mathcal{G}$ will not work.

We sketch a proof that there is a clustering problem $(\mathcal{X}, \mathcal{L}, \mathcal{I}, \mathcal{T}, \mathcal{D}, \mathcal{G})$ satisfying P_1 through P_6 such that for each number $k \geq 3$, the problem of finding for all instances $(X_n, L_n, I_n, T_n, D_n, G_n, S)$, a best clustering among those with at most k clusters, is NP-hard. Our example is a cluster analysis problem, thus for each n, $L_n = \{c : c$ is a finite

subset of $X_n\}$, and I_n is the identity function. As in all cluster analysis problems within our framework, properties P_1, P_2, and P_4 hold immediately.

Let $k \geq 3$ be given, and let $\mathcal{X}, \mathcal{T}, \mathcal{D}$, and $\mathcal{G}$ be as defined below:

$\mathcal{X} = \{X_{2v}\}$, where X_{2v} is the set of (even) length $2v$ strings. A given string of length $2v$ will represent a vertex in an undirected graph of v vertices if the first half of the string contains a single "1" bit . The single "1" among the first v bits indicates which vertex it is, and the remaining v bits give adjacency information with other vertices, i.e., a "1" in position $v + j$ indicates that the vertex is adjacent to vertex j. Note that any graph with v vertices may be represented by a finite set of points of X_{2v}, although not every finite subset of X_{2v} represents a graph. For example, the graph of 5 vertices with edges $(1,2),(1,3),(1,5),(2,3),(4,5),(3,5)$ corresponds to the subset of X_{10} given by the elements $\{x_1, x_2, x_3, x_4, x_5\}$ below: (a comma is inserted between the 5th and 6th bits, and each element is parenthesized to aid the interpretation.)

$x_1 = (10000, 01101)$ (vertex 1 adjacent to 2,3, and 5)
$x_2 = (01000, 10100)$ (vertex 2 adjacent to 1 and 3)
$x_3 = (00100, 11001)$ (vertex 3 adjacent to 1,2, and 5)
$x_4 = (00010, 00001)$ (vertex 4 adjacent to 5)
$x_5 = (00001, 10110)$ (vertex 5 adjacent to 1,3, and 4)

Given as input any even number $2v$ and finite subset S of X_{2v}, it is decidable in polynomial time whether S represents a *subset* of the vertices of some undirected graph of v vertices, or whether no undirected graph has a subset of vertices represented by the elements of S. (What must be checked is that (1) Each string of S has a single "1" among the first v bits; (2) For each $i \leq v$, there is at most one string in S with a single "1" in position i; and (3) If a string representing a vertex numbered i has a "1" in position $v + j$, then the string representing vertex j (if it appears in S) has a "1" in position $v + i$.)

For a clustering C, Let $T_{2v}(C) = 0$ if the union of the clusters of C is not a finite subset of X_{2v} representing a subset of the vertices of some undirected graph, OR if there is a cluster $c \in C$ such the representations of two vertices which are adjacent in the represented subgraph are both contained in c. Let $T_{2v}(C) = 1$ otherwise. In the example above,

$T_{10}(\{\{x_1, x_4\}, \{x_2, x_5\}\}) = 1$, since the elements of the clusters are consistent with some 5 vertex undirected graph, and no two adjacent vertices appear in any single cluster. On the other hand, $T_{10}(\{\{x_1, x_2\}, \{x_3\}\}) = 0$, since in any graph which contains the vertices x_1, x_2, and x_3, x_1 is adjacent to x_2 and they appear in the same cluster. As a final example, $T_{10}(\{\{1001\}, \{0100\}\}) = 0$, because the adjacency information between vertex 1 and 2 in the two vertex graph represented is inconsistent.

Let D_{2v} be the constant function $D_{2v}(C) = 2v$, and let $G_{2v}(C) = min(T_{2v}(C), D_{2v}(C))$.

Now it is easily verified that the clustering problem $(\mathcal{X}, \mathcal{L}, \mathcal{I}, \mathcal{T}, \mathcal{D}, \mathcal{G})$ satifies P_3, P_5 and P_6. Since this is a cluster analysis problem, it also satisfies P_1, P_2, and P_4.

We reduce the NP-hard graph k-colorability problem (Garey & Johnson, 1979) to the problem of finding a best solution among all clusterings having at most k clusters for the problem $(\mathcal{X}, \mathcal{L}, \mathcal{I}, \mathcal{T}, \mathcal{D}, \mathcal{G})$. For each $k \geq 3$, the graph k-colorability problem is to determine whether there is a coloring of the vertices of a graph using at most k colors, so that no two adjacent vertices have the same color. Given a graph $\mathcal{A} = (V, E)$, with v vertices, we form an instance $(X_{2v}, L_{2v}, I_{2v}, T_{2v}, D_{2v}, G_{2v}, S)$ of the clustering problem above by letting the set $S \subset X_{2v}$ to be clustered be exactly those elements of X_{2v} which represent vertices V of the graph $\mathcal{A}$ with adjacency information given by E. A simple argument shows that the graph $\mathcal{A}$ is k-colorable iff there is a clustering C for this instance with at most k clusters such that $G_{2v}(C) = 1$. (The clusters consist of representations of vertices to be colored with the same color.) Otherwise, any clustering for this instance with at most k clusters has $G_{2v}(C) = 0$. It follows that for each k, any algorithm for solving $(\mathcal{X}, \mathcal{L}, \mathcal{I}, \mathcal{T}, \mathcal{D}, \mathcal{G})$ by finding the best clustering with at most k clusters can be used to solve the graph k-colorability problem. We have thus proved

Theorem 4.5 *For all $k \geq 3$ there are clustering problems $(\mathcal{X}, \mathcal{L}, \mathcal{I}, \mathcal{T}, \mathcal{D}, \mathcal{G})$ satisfying properties P_1 through P_6 such that, unless $P = NP$, there does not exist a polynomial time algorithm for finding a best clustering among all clusterings with at most k clusters for every instance $(X_n, L_n, I_n, T_n, D_n, G_n, S)$.*

4.3 Variants of T and D

Although Theorem 4.1 shows that only very general assumptions on the functions $\mathcal{G}$ are needed, the results apply only when the functions $\mathcal{T}$ satisify property P_5, and the functions

$\mathcal{D}$ satisfy property P_3. Measures of tightness such as "density" of individual clusters allow the tightness to increase as a clustering is generalized, since "sparse" clusters may become "dense" when new points are added. Thus property P_5 is violated for this type of measure. Similarly, if the distance functions $\{D_n\}$ are defined as the *maximum* intercluster distance d_n, property P_3 is no longer satisfied. In this section we show that (assuming $P \neq NP$), P_5 is necessary in the sense that properties P_1, P_2, P_3, P_4, and P_6 alone are not sufficient for a clustering problem to be solvable in polynomial time. We conclude by observing that if the functions $\{D_n\}$ are in fact the maximum intercluster distance, then the clustering problem is trivial (assuming P_5 still holds).

Theorem 4.6 *There is a clustering problem $(\mathcal{X}, \mathcal{L}, \mathcal{I}, \mathcal{T}, \mathcal{D}, \mathcal{G})$ satisfying P_1, P_2, P_3, P_4, and P_6 that is not solvable in polynomial time unless $P = NP$.*

Proof: We need only reduce INDEPENDENT SET, an NP-hard problem (Garey & Johnson, 1979), to a cluster analysis problem satisfying properties P_3 and P_6. An instance of INDEPENDENT SET is a graph $\mathcal{A} = (V, E)$, and a positive integer $k \leq |V|$. The problem is to determine if $\mathcal{A}$ contains an independent set of size k or more, i.e., a subset $V' \subseteq V$ such that $|V'| \geq k$ and such that no two vertices of V' are joined by an edge in E.

Let the clustering problem $(\mathcal{X}, \mathcal{L}, \mathcal{I}, \mathcal{T}, \mathcal{D}, \mathcal{G})$ be defined as in the proof of Theorem 4.5, except that T_{2v} has the modified definition given by: $T_{2v}(C) = 0$ if the union of the clusters in C is not a finite subset of X_{2v} representing *all* of the vertices of some undirected graph with v vertices, OR if there is a cluster $c \in C$ such the representations of two vertices which are adjacent in the represented graph are both contained in c. $T_{2v}(C) =$ the number of elements in the largest cluster of C otherwise.

$(\mathcal{X}, \mathcal{L}, \mathcal{I}, \mathcal{T}, \mathcal{D}, \mathcal{G})$ satisfies properties P_1, P_2, P_3, P_4, and P_6, since only the definition of $\mathcal{T}$ has been changed from the proof of Theorem 4.5, and property P_5 has been dropped. Also note that for each C, $D_{2v}(C) = 2v \geq T_{2v}(C)$, so $G_{2v}(C) = T_{2v}(C)$. It is now easily shown that a graph $\mathcal{A}$ has an independent set of size k iff the instance $(X_{2v}, L_{2v}, I_{2v}, T_{2v}, D_{2v}, G_{2v}, S)$, with S representing the graph $\mathcal{A}$, has a solution C with $G_{2v}(C) = k$. □

Finally, suppose that we modify part (a) of property P_3 so that D_n is now the *maximum* inter-cluster distance $d_n(c, c')$ among all clusters $c, c' \in C$, where d_n satisfies part (b) of property P_3. Then any clustering problem $(\mathcal{X}, \mathcal{L}, \mathcal{I}, \mathcal{T}, \mathcal{D}, \mathcal{G})$ satisfying properties P_2, P_5, P_6, and this modified definition of P_3, is trivially solvable: The clustering given by C_1 in algorithm

A must be a best, since it is a specialization of every best clustering; tightness cannot increase under generalization, *nor can distance*, by the modified property P_3. Thus G_n cannot increase either.

5 Conclusion

The main results in this paper can be summarized as follows:

- The agglomerative algorithm will find a best (conceptual) clustering of a set of points if the similarity measure (for clusterings) and the dissimilarity (between clusters) are monotone with respect to generalization, the objective function is monotone with respect to similarity and dissimilarity, and the language is tractable. Tractable in this case means that the clusterings of the language are not too large, that it is possible to efficiently determine whether a point is in a cluster, and that there exists and it is possible to find the most specific clustering in the language satisfying certain conditions. The "identity" language for cluster analysis trivially has these properties.

- Under these same conditions, the agglomerative algorithm will find a best clustering with *at least* k clusters for any fixed k less than the size of the sample set being clustered.

- Finding the best clustering with *at most* k clusters is NP-hard under these conditions.

- If the measure of similarity is not monotone with respect to generalization, then finding an optimal clustering is NP-hard, even if the other monotone properties and the language properties are satisfied.

These results have several interesting implications. First, the agglomerative algorithm is more widely applicable than would be expected from such a simple technique. Under straightforward and intuitively natural conditions on the metric, the objective function, and the cluster description language, it finds a best clustering in polynomial time. The language restrictions are satisfied by conjunctive, attribute-based languages, including those using internal disjunction. They also apply to several interesting geometric languages. They do not hold for the existentially-quantified conjunctive predicate calculus statements sometimes used to represent structured objects (Stepp, 1987a; Larson, 1977).

Finally, as would be expected, it seems that finding a best clustering with a given number of clusters is hard. The implication is that clustering algorithms which try to find a best clustering of a certain size will have to be content with sub-optimal results. It also confirms the intuition that heuristic techniques and domain knowledge are probably necessary to produce good solutions.

We would like to extend the results to metrics that, for example, include notions such as density or average similarity over clusters. Additionally, it would be useful to be able to weaken the restrictions on distance (for a clustering) so that it is not the minimum inter-cluster distance.

A problem we have not addressed here is the notion of predictive clustering, along lines of learnability as described by Valiant (1984), and Blumer, Ehrenfeucht, Haussler, & Warmuth (1986). (See also (Kearns, Li, Pitt, & Valiant, 1987).) The idea is to develop a clustering which is "good" for an entire space X (under an unspecified probability distribution), given only randomly generated points from X. We have definitions that seem suitable for this problem, and have some preliminary results indicating that this is a significantly harder problem than nonpredictive clustering. These results may be presented in a future paper.

Acknowledgements

We are grateful to the referee for two thorough reviews and for simplifying the proof of Theorem 4.1 while extending its scope. L. Pitt was supported by the Department of Computer Science, University of Illinois at Urbana-Champaign, and R. E. Reinke was supported in part by the National Science Foundation under grant NSF IST-85-11170 and in part by the Office of Naval Research under grant N00014-82-K-0186.

References

Anderberg, M. (1973). *Cluster Analysis for Applications.* New York: Academic Press.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, A. (1986). Classifying learnable geometric concepts with the Vapnik-Chervonenkis dimension. *Proceedings of the* 18^{th} *Annual ACM Symposium on Theory of Computation* (pp 273-282). Berkeley, CA: Association for Computing Machinery.

Carbonell, J.G., Michalski, R.S. & Mitchell, T.M. (1983). An Overview of Machine Learning. In R.S. Michalski, J.G. Carbonell & T.M. Mitchell (Eds.), *Machine Learning: An Artifical Intelligence Approach.* Palo Alto, CA: Tioga.

Cohen, P.R. & Feigenbam, E.A. (Eds.). (1982). *The Handbook of Artificial Intelligence*, Los Altos, CA: William Kaufmann.

Duda, R., & Hart, P. (1973). *Pattern classification and scene analysis.* New York: John Wiley & Sons.

Edelsbrunner, H. (1987). *Algorithms in Combinatorial Geometry.* W. Brauer, G. Rozenberg, A. Salomaa (Eds.), Monographs in Theoretical Computer Science (Vol. 10). Heidelberg: Springer-Verlag.

Everitt, B. (1980). *Cluster Analysis.* London: Heinemann Educational Books.

Fisher, D. (1985). A Proposed Method of Conceptual Clustering for Structured and Decomposable Objects. *Proceedings of the Third International Machine Learning Workshop* (pp. 38-40). Skytop, PA.

Fisher, D. (1987). Improving Inference Through Conceptual Clustering. *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 461-465). Seattle, WA: Morgan Kaufmann.

Fisher, D. & Langley, P. (1985). Approaches to Conceptual Clustering. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 691-697). Los Angeles, CA: Morgan Kaufmann.

Garey, M., & Johnson, D. (1979). *Computers and intractability: a guide to the theory of NP-completeness.* San Francisco, CA: W. H. Freeman.

Hartigan, J. (1975). *Cluster Algorithms.* New York: John Wiley & Sons.

Kearns, M., Li, M., Pitt, L., & Valiant, L. G. (1987). Recent Results in Boolean Concept Learning. *Proceedings of the* 4^{th} *International Machine Learning Workshop* (pp. 337-352). Irvine, CA: Morgan Kaufmann.

Larson, J. (1977). *Inductive Inference in the Variable-Valued Predicate Logic System VL21: Methodology and Computer Implementation.* Ph.D. Thesis, Department of Computer Science, University of Illinois, Urbana, Illinois.

Lebowitz, M. (1983). Generalization from Natural Language Text. *Cognitive Science, 7*, 1-40.

Michalski, R.S. (1980). Knowledge Acquisition Through Conceptual Clustering: A Theoretical Framework and an Algorithm for Partitioning Data into Conjunctive Concepts. *International Journal of Policy Analysis and Information Systems, 4*, 219-243.

Michalski, R.S. (1983). A Theory and Methodology of Inductive Learning. In R. S. Michalski, J. G. Carbonell & T. M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach*, Palo Alto, CA: Tioga.

Michalski, R.S. & Stepp, R.E. (1983) Learning from Observation: Conceptual Clustering. In R.S. Michalski, J.G. Carbonell & T.M. Mitchell (Eds.), *Machine Learning: An Artifical Intelligence Approach*, Palo Alto, CA: Tioga.

Mogenson, B. (1987). *Goal-Oriented Conceptual Clustering: The Classifying Attribute Approach.* Master's thesis, Department of Electrical and Computer Engineering, University of Illinois, Urbana, IL.

Rendell, L.A. (1983). Toward a Unified Approach for Conceptual Knowledge Acquisition. *AI Magazine, 4*, 19-27.

Romesburg, H. (1984). *Cluster Analysis for Researchers*, Belmont, CA: Lifetime Learning.

Stepp, R. (1984). *Conjunctive Conceptual Clustering: A Methodology and Experimentation.* Ph.D. Thesis, Department of Computer Science, University of Illinois, Urbana, IL.

Stepp, R. (1987a). Machine Learning from Structured Objects. *Proceedings of the* 4^{th} *International Machine Learning Workshop* (pp. 353-363). Irvine, CA: Morgan Kaufmann.

Stepp, R. (1987b). Concepts in Conceptual Clustering. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* (pp. 211-213). Milan, Italy: Morgan Kaufmann.

Stepp, R. & Michalski, R.S. (1986). Conceptual Clustering: Inventing Goal-Oriented Classifications of Structured Objects. In R.S. Michalski, J.G. Carbonell & T.M. Mitchell (Eds.), *Machine Learning: An Artifical Intelligence Approach (Vol. 2)*. Los Altos, CA: Morgan Kaufmann.

Valiant, L. G. (1984). A theory of the learnable. *Commumnications of the ACM, 27*, 1134-1142.